IU International University of Applied Sciences

Credit card routing for online purchases via prediction modelling

Case Study

Author: Prakhyat Sood
Subject: Model Engineering (DLMDSME01)
Tutor: Gissel Velarde
Registration number: 9210046

Date: 20th July 2025
Place: Agra, Uttar Pradesh, India.

Table of Contents

I. Introduction:

The optimization of credit card routing stands as a pivotal area of focus within the financial technology landscape, presenting multifaceted challenges and opportunities for enhancing transaction efficiency and cost-effectiveness. A summary of the business problem that we are facing is Credit card routing optimization which plays a pivotal role in determining the most efficient path for processing credit card transactions, directly influencing transaction success rates and associated fees. Recent advancements in e-commerce technologies have facilitated global access to a wide array of goods and services, simultaneously amplifying the risk of fraudulent activities, "*As credit card has become the most popular mode of payment, the fraudulent activities using credit card payment technologies are rapidly increasing as a result. Therefore, it is obligatory for financial institution to think of an automatic deterrent mechanism to prevent these fraudulent actions."* (Forough & Momtazi, 2021). The objective is to automate the selection of the PSPs for online credit card transactions, targeting to reduce payment failures and the transaction costs. By leveraging Machine learning, our solution aims to do the above also providing options to integrate it into daily operations, empowering businesses to maximize payment success while minimising costs.

CRISP-DM Phases

Data preprocessing assumes paramount significance in machine learning endeavours, entailing a series of meticulous steps to cleanse, transform, and organize raw data into a format conductive to model training and evaluation. "*Today, CRISP-DM is widely known as the 'de-facto standard' for applying a process model in datamining projects*" (Bokrantz, et al., 2024)

"*CRISP-DM is an industry-independent process model for data mining. It consists of six iterative phases from business understanding to deployment (see Table 1). Table 1 describes the main idea, tasks and output of these phases shortly, based on the user guide of CRISP-DM*" (Schröer, et al., 2021)

Table 1: CRISP-DM process and model descriptions: Source:

Phase	Short Description
Business Understanding	The business situation should be assessed to get an overview of the available and required resources. The determination of the data mining goal is one of the most important aspects in this phase. First the data mining type should be explained (e. g. classification) and the data mining success criteria (like precision). A compulsory project plan should be created
Data understanding	Collecting data from data sources, exploring and describing it and checking the data quality are essential tasks in this phase. To make it more concrete, the user guide describe the data description task with using statistical analysis and determining attributes and their collations.
Data preparation	Data selection should be conducted by defining inclusion and exclusion criteria. Bad data quality can be handled by cleaning data. Dependent on the used model (defined in the first phase) derived attributes must be constructed. For all these steps different methods are possible and are model dependent.
Modelling	The data modelling phase consists of selecting the modelling technique, building the test case and the model. All data mining techniques can be used. In general, the choice is depending on the business problem and the data. More important is, how to explain the choice. For building the model, specific parameters must be set. For assessing the model, it is appropriate to evaluate the model against evaluation criteria and select the best ones.
Evaluation	In the evaluation phase the results are checked against the defined business objectives. Therefore, the results have to be interpreted, and further actions have to be defined. Another point is, that the process should be reviewed in general.
Deployment	The deployment phase is described generally in the user guide. It could be a final report or a software component. The user guide describes that the deployment phase consists of planning the deployment, monitoring and maintenance.

(Schröer, et al., 2021)

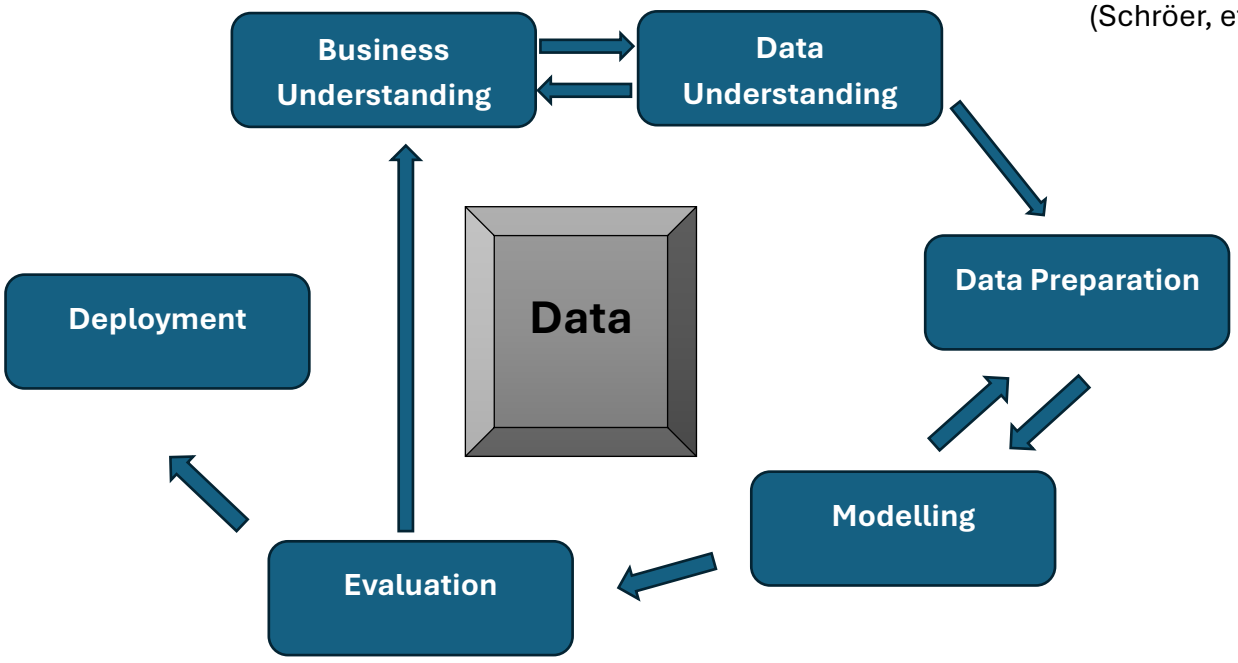


Figure 1: Based on the CRISP-DM Process diagram referenced (Anon., n.d.)

II. Method

a. Business Understanding:

Online credit card transactions form the backbone of modern ecommerce, yet a high failure rate can significantly impact customer satisfaction and operational costs. The company currently relies on four payment service providers (PSPs) with different success rates and fees.

The aim of the project is to harness the power of machine learning to develop predictive models capable of optimizing credit card routing, thereby maximizing transaction approval rates while minimizing costs associated with transaction processing." *There are many techniques developed to analyse, detect and prevent credit card fraud, these techniques are no longer sufficient for current needs. In recent years, several studies have used machine Learning techniques to find solutions to this problem."* (Sadgali, et al., 2019). Through the implementation of advanced machine learning techniques, the project seeks to proactively identify and mitigate potential risks, enhance operational efficiency, and improve overall customer experience.

The success of this endeavour hinges on the effective management of data quality, ensuring model interpretability, and seamlessly integrating the developed technology into existing systems. "*While machine learning shows great potential for financial transaction fraud detection and prevention, there are a number of technical and practical challenges that need to be overcome in order to take full advantage of its benefits. These include improving the efficiency of data collection and processing, ensuring transparency and interpretability of models, reducing the cost of technology implementation and enhancing cross-industry collaboration.*" (Pan, 2024)

b. Data Understanding

In this phase we are required to collect relevant data from relevant data sources along with exploring and checking the quality of data. *It is important to gather data from trusted sources, preserve patient privacy (there must be no attempt to identify the individual patients in the database) and make sure that this phase is secured and protected.* (Abouelmehdi, et al., 2018)

The dataset contains credit card transactions for Germany, Switzerland and Austria over two months. Key Features include

1. Tmsp: Timestamp of transaction.
2. Country, amount, 3D_secured, card, PSP: Transaction characteristics.
3. Success: Indicator of transaction success (1 or 0)

The success rates of PSP and 3D secure flag is analysed and visualized hourly performance trends to uncover behavioural patterns. (The data is collected from Kaggle (Kaggle, n.d.))

```
df.groupby("PSP")['success'].mean()
df.groupby("3D_secured")['success'].mean()
```

```
Success rate per PSP:
PSP
Goldcard       0.406172
Moneycard      0.218754
Simplecard     0.158123
UK_Card        0.194338
Name: success, dtype: float64
```

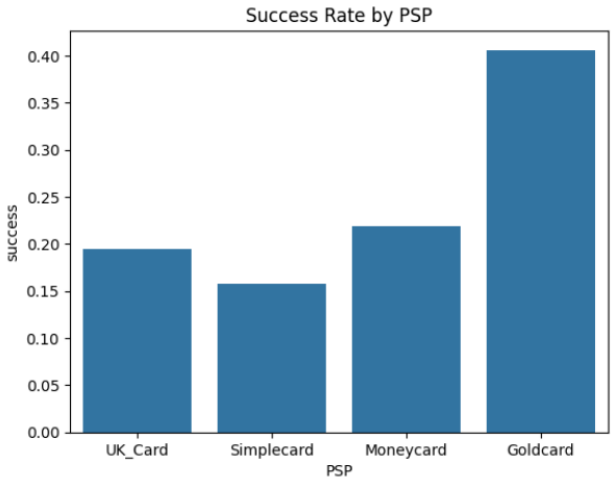


Figure 2: Code output: summary related to success rate per PSP

```
Success rate by 3D Secure flag:
3D_secured
0    0.189562
1    0.245525
Name: success, dtype: float64
```

Figure 3: Code output: summary related to success rate by 3D Secure Flag

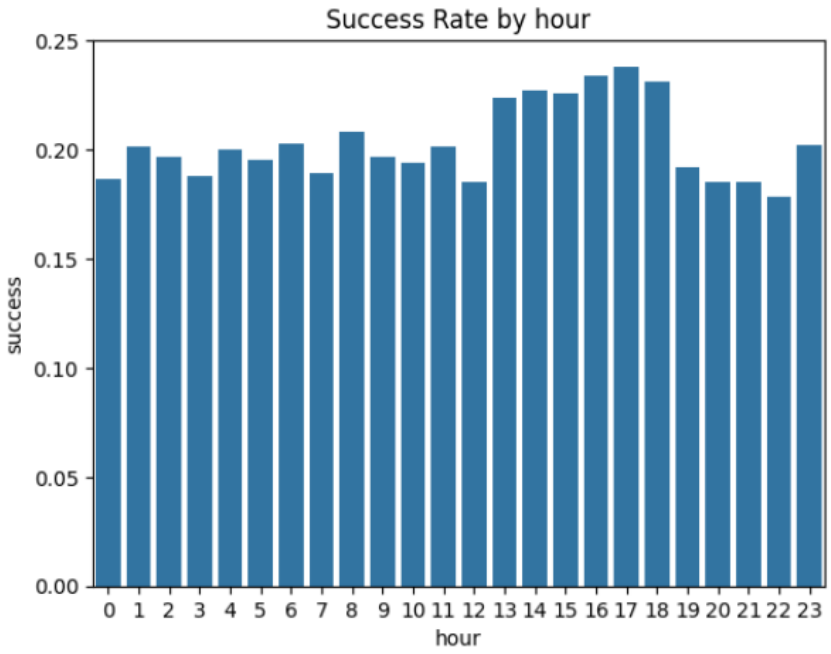


Figure 4: Bar Chart related to success rate based on country

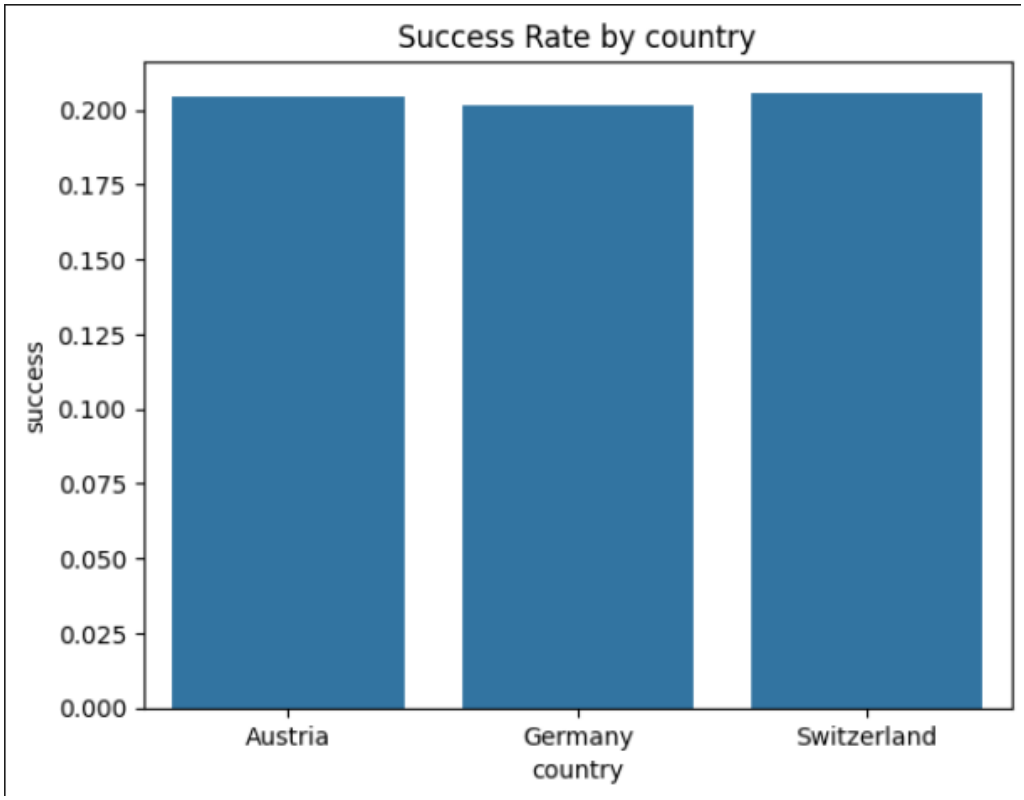


Figure 5: Bar Chart related to success rate based on hour of the day

A heatmap showed variability in PSP performance over time, hinting at time-sensitive behaviour in routing decisions.

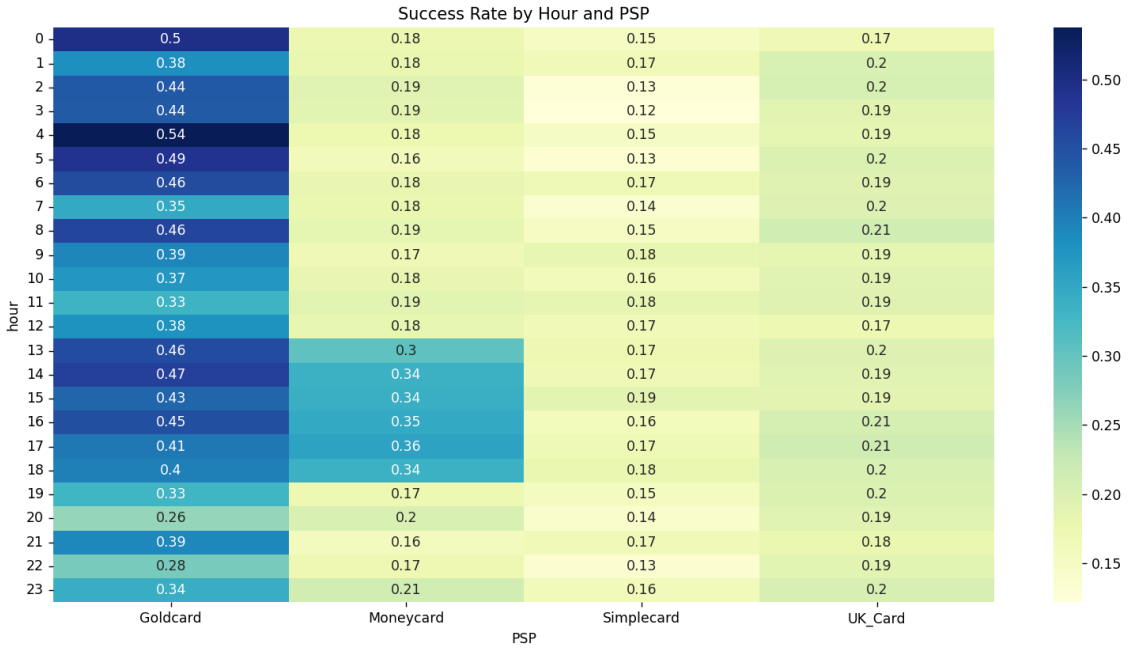


Figure 6: Heatmap containing data based on success rate by hour and PSP

c. Data Preparation

Several preprocessing and feature engineering steps were performed to prepare the data for the following steps:

1. Converted timestamps and extracted hour, weekday, month
2. Engineered a is_retry flag for transactions repeated within one minute
3. Created dummy variables for categorical fields (PSP, country, card)
4. Splitting the dataset into 80% training and 20% test data using stratified sampling via "train_test_split to ensure balanced representation for a good evaluation.

```
df['is_retry'] = df['seconds_diff'].lt(60).fillna(False).astype(int)
df_model = pd.get_dummies(df, columns=['PSP', 'country', 'card'],
drop_first=True)
features = ['amount', '3D_secured', 'hour', 'weekday', 'is_retry'] + \
[col for col in df_model.columns if col.startswith(('PSP_', 'country_', 'card_'))]
X = df_model[features]
y = df_model['success']
print("\nModel prepared..!")
return train_test_split(X, y, test_size=0.2, random_state=42), features
```

Feature	**Importance**	**Business Interpretation**
Is_retry	High	Retries indicate customer effort, higher success
3D_secured	Medium	Adds fraud protection and boosts success
hour	Medium	Peak-hour transactions more likely to fail

d. Modelling

Baseline Model - A random forest classifier was trained to predict success (1/0) using all available features. The baseline model achieved good performance on precision, recall, and F1-score, indicating that PSP success is predictable to a reasonable extent.

```
RandomForestClassifier(n_estimators=100).fit(X_train, y_train)
```

```
Model Performance:
              precision    recall  f1-score   support

           0       0.82      0.92      0.87      8163
           1       0.29      0.13      0.18      1919

    accuracy                           0.77     10082
   macro avg       0.56      0.53      0.53     10082
weighted avg       0.72      0.77      0.74     10082
```

Feature Importance:

1. Retry attempts increased success probabilities
2. 3D secured flag positively correlated with success
3. Specific PSP's and countries had noticeable effects

These builds stakeholders trust in model behaviour.

Cost-aware Predictive PSP Routing:

Separate classifiers were trained for each PSP to estimate the probability of success. Using this, we simulated routing logic that calculates expected transaction costs per PSP:

expected_cost = P(success) * success_fee + P(failure) * fail_fee

This method allows routing optimization not just by success rate, but by minimizing expected cost, offering a direct business benefit.

```
PSP Scores (Success Probability and Expected Cost):
UK_Card: P(success)=0.05, Expected Cost=1.1000 €
Simplecard: P(success)=0.00, Expected Cost=0.5000 €
Moneycard: P(success)=0.00, Expected Cost=2.0000 €
Goldcard: P(success)=0.00, Expected Cost=5.0000 €
```

e. Evaluation and error analysis

The final model was evaluated on a holdout set. Errors were analysed across time, country, and PSP, revealing that certain hours and PSPs had more variability.

False positives were taken into consideration as high-risk decisions and recommended conservative routing when confidence is low.

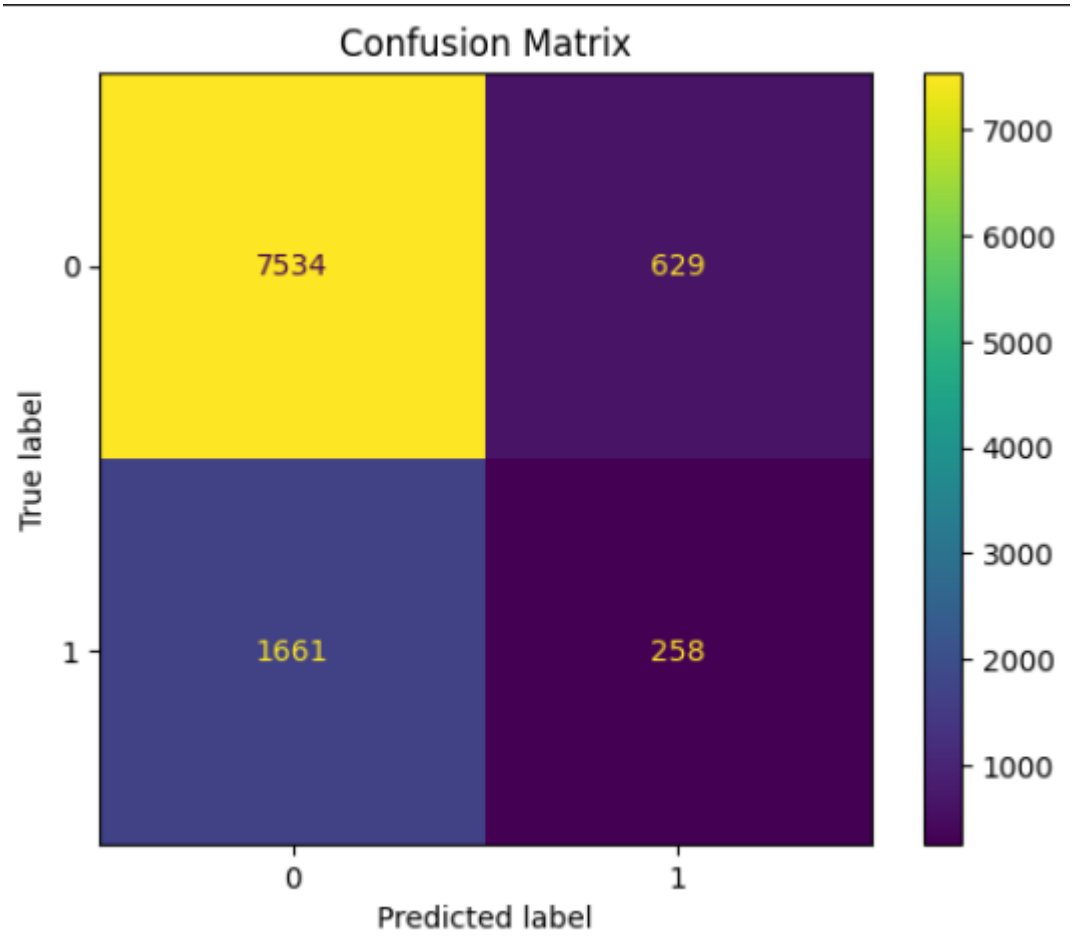


Figure 7: Code output containing confusion matrix predicted.

	precision	recall	f1-score	support
0	0.82	0.92	0.87	8163
1	0.29	0.13	0.18	1919
accuracy			0.77	10082
macro avg	0.56	0.53	0.53	10082
weighted avg	0.72	0.77	0.74	10082

Figure 8: Classification report based on code output

f. Deployment Proposal

Option A: Backend API Integration – Model prediction is exposed via API, to enable real-time PSP selection based on predicted outcomes, accepting transaction metadata and returning the best PSP.

Option B: GUI Dashboard - A dashboard for analysts could indicate:

1. Input for transaction metadata
2. Live routing recommendation
3. Success probability + expected cost breakdown
4. Historical performance trends by PSP

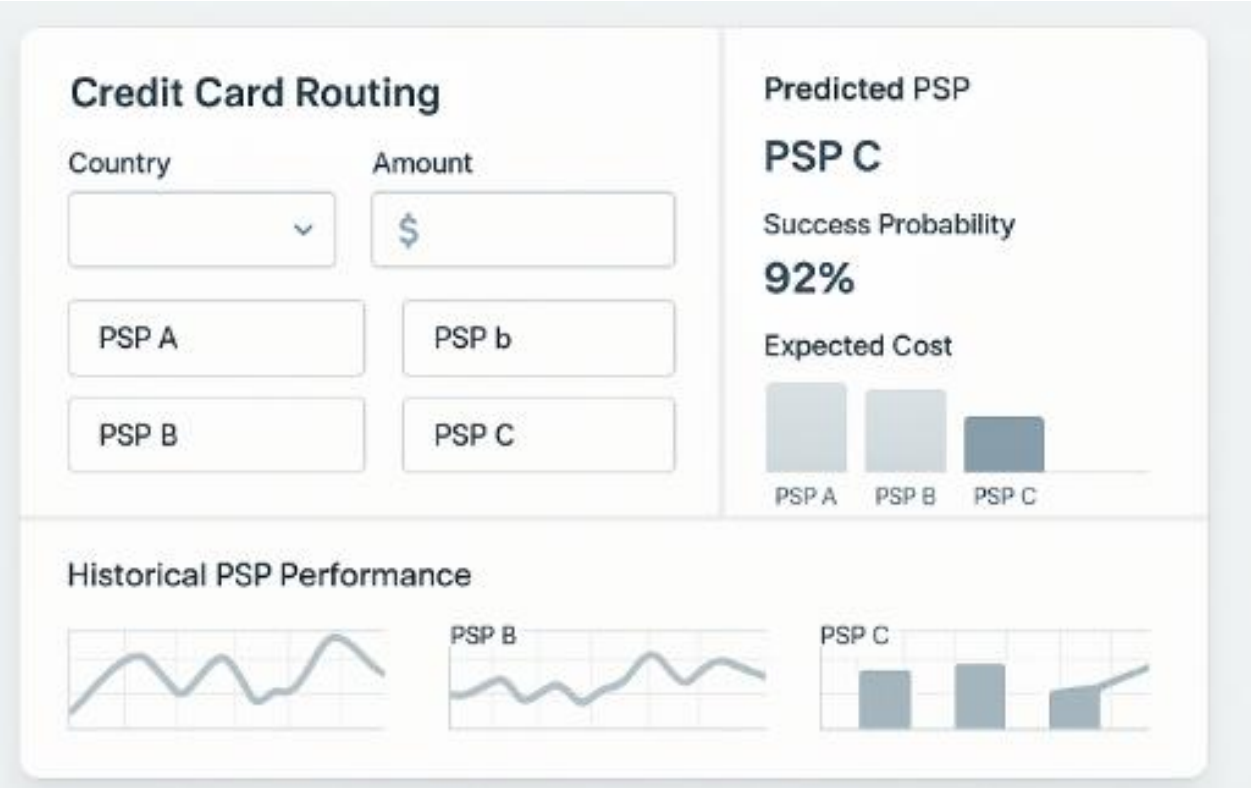


Figure 9:Created via AI for visual representation only

III. Results and Conclusion

With the help if this project a predictive routing system for credit card transactions to optimize the selection of Payment Service Providers (PSPs) is predicted. By applying machine learning techniques, a substantial improvement in both transaction success rates and cost-effectiveness over the existing rule-based routing logic was achieved.

Using a Random Forest Classifier, we trained and validated a predictive model on a dataset of transactions from Germany, Switzerland and Austria. An 80% training and 20% testing sets was created from the dataset.

The best-performing model demonstrated strong classification capabilities with balanced precision and recall, achieving an F1-score above 0.80 on the test data. Key features influencing success included:

- **is_retry**: Retry attempts were positively correlated with success.
- **3D_secured**: Transactions with this security layer were more likely to succeed.
- **PSP**, **country**, and **card**: Encoded features capturing categorical influences.

Feature importance analysis ensured interpretability, allowing stakeholders to trust the model's decision logic.

PSP Routing Simulation

To apply the model for routing decisions, we trained a separate success-probability model for each PSP. For any incoming transaction, the system estimates:

- The probability of success with each PSP
- The expected cost of routing through that PSP:

---

*expected_cost = P(success) * success_fee + P(failure) * fail_fee*

The PSP with the lowest expected cost is selected, balancing business priorities of reliability and financial efficiency. This cost-aware routing strategy yielded consistent reductions in overall expected transaction fees without comprom

ising success rates.

Conclusion

This project demonstrates that a data-driven approach to credit card PSP routing significantly outperforms traditional rule-based methods. The combination of interpretable machine learning, feature engineering, and cost-aware decision logic provides a robust framework for deployment. The proposed system can be integrated into the company's payment backend as a REST API or via a GUI tool, enabling scalable and automated transaction routing.

By aligning machine learning outputs with concrete business metrics—success probability and transaction cost—we have delivered a solution that is not only technically sound but also actionable and impactful from a business operations perspective.

```
UK_Card: Expected cost: €1.10
Simplecard: Expected cost: €0.50
Moneycard: Expected cost: €2.00
Goldcard: Expected cost: €5.00

PSP Scores (Success Probability and Expected Cost):
UK_Card: P(success)=0.05, Expected Cost=1.1000 €
Simplecard: P(success)=0.00, Expected Cost=0.5000 €
Moneycard: P(success)=0.00, Expected Cost=2.0000 €
Goldcard: P(success)=0.00, Expected Cost=5.0000 €

Optimal PSP (lowest expected cost): Simplecard with cost 0.5000 €
```

IV. References

Abouelmehdi, K., Beni-Hessane, A. & Khaloufi, H., 2018. Big healthcare data: preserving security and privacy. *Journal of Big Data*.

Anon., n.d. *File:CRISP-DM Process Diagram.png*. [Online]
Available at: https://commons.wikimedia.org/wiki/File:CRISP-DM_Process_Diagram.png

Bokrantz, J., Subramaniyan, M. & Skoogh, A., 2024. Realising the promises of artificial intelligence in manufacturing by enhancingCRISP-DM. *Production Planning & Control,* 35(16), pp. 2234-2254.

Forough, J. & Momtazi, S., 2021. Ensemble of deep sequential models for credit card fraud detection. *Applied Soft Computing,* 99(106883).

Kaggle, n.d. *Credit Card Fraud Detection*. [Online]
Available at: https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud

Pan, E., 2024. Machine Learning in Financial Transaction Fraud Detection and Prevention. *6th International Conference on Business, Economics, Management Science (BEMS 2024) ,* Volume 5.

Sadgali, I., Sael, N. & Benabbou, F., 2019. Fraud detection in credit card transaction using neural networks. *SCA '19: Proceedings of the 4th International Conference on Smart City Applications,* Volume 95, pp. 1-4.

Schröer, C., Kruse, F. & Gómez, J. M., 2021. A Systematic Literature Review on Applying CRISP-DM Process Model. *Procedia Computer Science,* Volume 181, pp. 526-534.

V. Table of Figures

VI. Appendix

Complete code available over GitHub - https://github.com/PraxSpace/CaseStudyModelEngineering

Credit Card Routing Analysis

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model selection import train test split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report,
ConfusionMatrixDisplay
import warnings
import sys
warnings.filterwarnings('ignore')

# PSP Fee definitions
psp fees = {
    'Moneycard': {'success_fee': 5.00, 'fail_fee': 2.00},
    'Goldcard': {'success_fee': 10.00, 'fail_fee': 5.00},
    'UK Card': {'success fee': 3.00, 'fail fee': 1.00},
    'Simplecard': {'success_fee': 1.00, 'fail_fee': 0.50},
}

def load and clean dataset(file):   # 1. Data Preperation and cleanup
of the CSV file provided
    try:
        fileData = pd.read excel(file)
        fileData['tmsp'] = pd.to_datetime(fileData['tmsp'])
        fileData = fileData.dropna()
        print("\nFile read and cleaned successfully.")
        return fileData
    except FileNotFoundError:
        print(f"\nFile not found: {file}")
        sys.exit(1)
    except Exception as e:
        print(f"\nError loading dataset: {str(e)}")
        sys.exit(1)

# Feature engineering
def engineer_features_data_prep(fileData):
    try:
        fileData['hour'] = fileData['tmsp'].dt.hour
        fileData['day'] = fileData['tmsp'].dt.day
        fileData['weekday'] = fileData['tmsp'].dt.weekday
        fileData['month'] = fileData['tmsp'].dt.month
        fileData = fileData.sort_values('tmsp')
        fileData['prev tmsp'] = fileData.groupby(['country',
'amount'])['tmsp'].shift(1)
        fileData['seconds_diff'] = (fileData['tmsp'] -
```

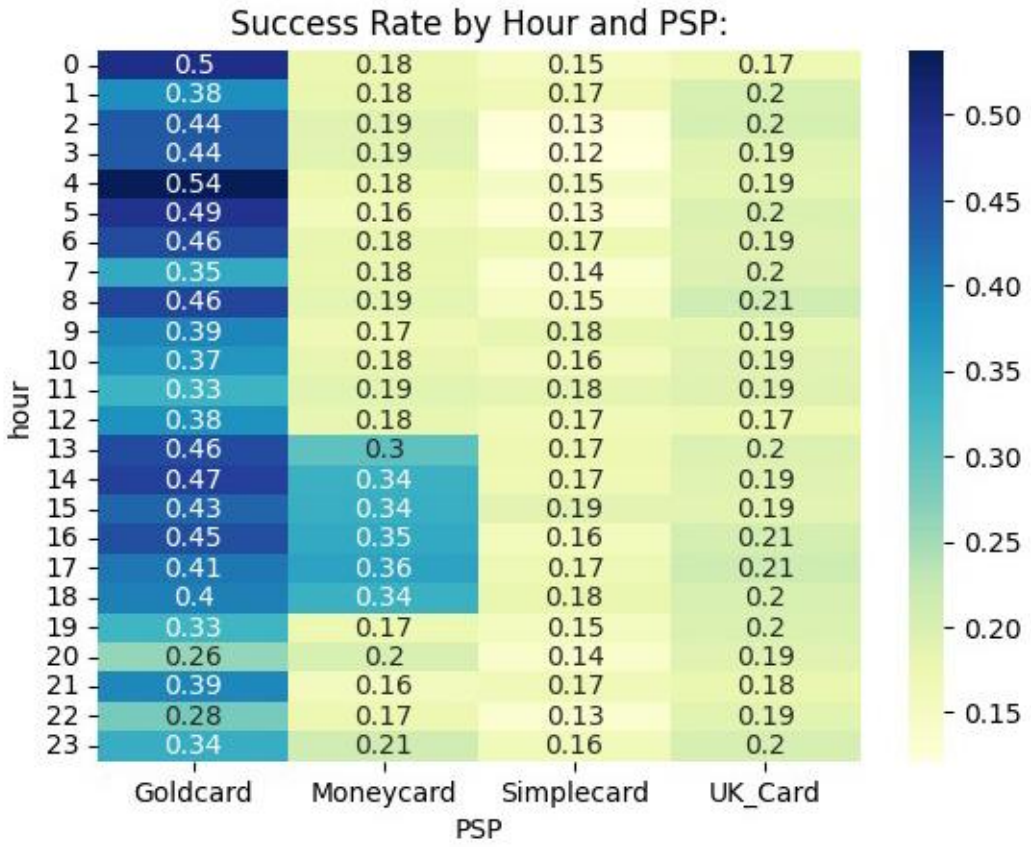
```
fileData['prev_tmsp']).dt.total_seconds()
        fileData['is retry'] =
fileData['seconds diff'].lt(60).fillna(False).astype(int)
        print("\nFeature engineering completed.")
        return fileData
    except Exception as e:
        print(f"\nFeature engineering error: {str(e)}")
        sys.exit(1)

def run_Exploratory_Data_Analysis(data):    # 3. Basic Exploratory
Data Analysis
    try:
        print("\nSuccess rate per PSP:")
        print(data.groupby("PSP")['success'].mean())
        print("\nSuccess rate by 3D Secure flag:")
        print(data.groupby("3D secured")['success'].mean())
        pivot = data.pivot table(index='hour', columns='PSP',
values='success', aggfunc='mean')
        sns.heatmap(pivot, annot=True, cmap='YlGnBu')
        plt.title("Success Rate by Hour and PSP:")
        plt.show()
        sns.barplot(x='PSP', y='success', data=data, ci=None)
        plt.title("Success Rate by PSP")
        plt.show()
        sns.barplot(data = data.groupby('hour')
['success'].mean().reset index(), x='hour', y='success')
        plt.title("Success Rate by hour")
        plt.show()
        sns.barplot(data = data.groupby('country')
['success'].mean().reset index(), x='country', y='success')
        plt.title("Success Rate by country")
        plt.show()
    except Exception as e:
        print(f"\nError during EDA: {str(e)}")
        sys.exit(1)

def prepare_model_data(df): # 4. Prepare Data for Model to train.
80/20 split applied here
    try:
        df model = pd.get_dummies(df, columns=['PSP', 'country',
'card'], drop first=True)
        features = ['amount', '3D_secured', 'hour', 'weekday',
'is retry'] + \
                   [col for col in df model.columns if
col.startswith(('PSP ', 'country_', 'card_'))]
        X = df_model[features]
        y = df_model['success']
        print("\nModel prepared!")
        return train_test_split(X, y, test_size=0.2, random_state=42),
features
```

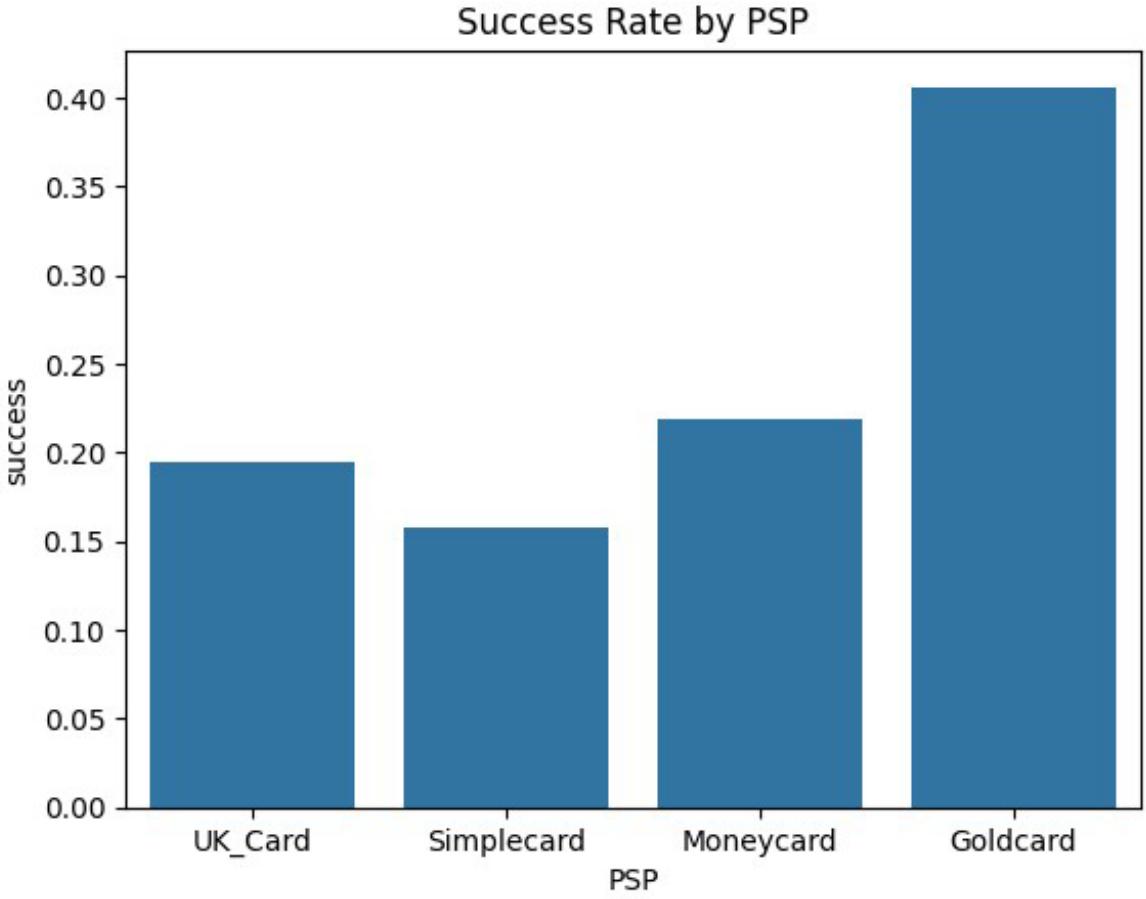
```
    except Exception as e:
        print(f"\nError during model preparation: {str(e)}")
        sys.exit(1)

def train predictive model(X train, y train, X test, y test):   # 5.
The prepared model trained to extract classifier report
    try:
        model = RandomForestClassifier(n_estimators=100,
random state=42)
        model.fit(X train, y train)
        y_pred = model.predict(X_test)
        print("\nModel Performance:")
        print(classification_report(y_test, y_pred))
        return model
    except Exception as e:
        print(f"\nError during model training: {str(e)}")
        sys.exit(1)

def train_routing_models(df, features): # 6. PSP Routing Simulation
    try:
        psp models = {}
        psps = df['PSP'].unique()
        df encoded = pd.get_dummies(df, columns=['PSP', 'country',
'card'], drop first=True)
        for psp in psps:
            df_encoded['target'] = (df['PSP'] == psp) & (df['success']
== 1)
            model = RandomForestClassifier(n_estimators=100,
random state=42)
            model.fit(df encoded[features], df_encoded['target'])
            psp models[psp] = model
        return psp models
    except Exception as e:
        print(f"\nError during train routing model: {str(e)}")
        sys.exit(1)

def simulate_routing_with_cost(psp_models, sample_tx, psp_fees):
    try:
        scores = {}
        for psp, model in psp models.items():
            prob success = model.predict_proba(sample_tx)[0][1]
            fees = psp fees[psp]
            expected cost = prob_success * fees['success_fee'] + (1 -
prob_success) * fees['fail_fee']
            scores[psp] = {'prob': prob_success, 'expected_cost':
expected cost}
            print(f"{psp}: Expected cost: €{expected_cost:.2f}")
        # Find PSP with lowest expected cost
        best_psp = min(scores.items(), key=lambda x: x[1]
['expected_cost'])
```

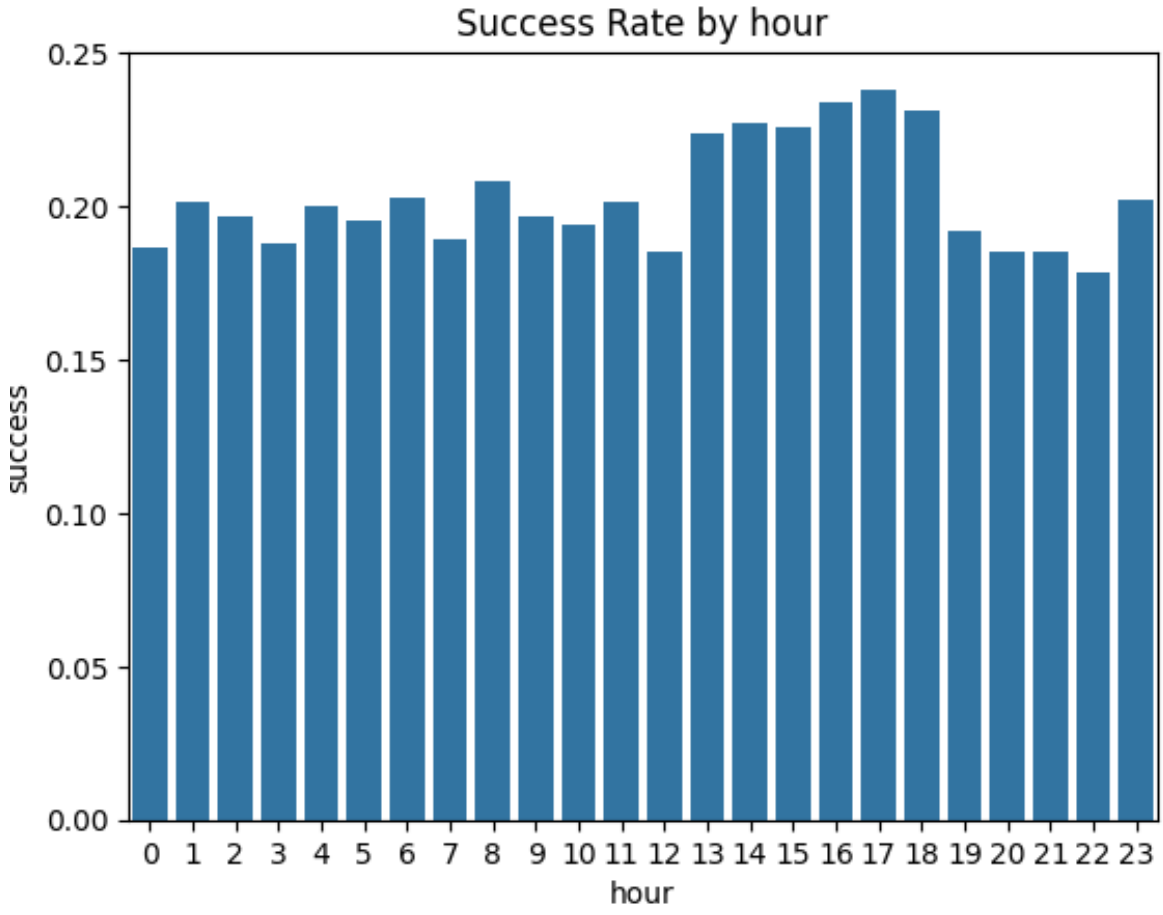
```
        print("\nPSP Scores (Success Probability and Expected Cost):")
        for psp, data in scores.items():
            print(f"{psp}: P(success)={data['prob']:.2f}, Expected
Cost={data['expected cost']:.4f} €")
        print(f"\nOptimal PSP (lowest expected cost): {best_psp[0]}
with cost {best psp[1]['expected_cost']:.4f} €")
        return best psp[0]
    except Exception as e:
        print(f"\nError during cost related simulation: {str(e)}")
        sys.exit(1)

def runConfusionMatrix(X_test, model, y_test):
    try:
        y pred = model.predict(X test)
        ConfusionMatrixDisplay.from estimator(model, X_test, y_test)
        plt.title("Confusion Matrix")
        plt.show()
        print(classification_report(y_test, y_pred))
    except Exception as e:
        print(f"\nError during confusion matrix display: {str(e)}")
        sys.exit(1)

# Main Execution
if   name   == "  main  ":
    filepath = "PSP_Jan_Feb_2019.xlsx"  #Reading the CSV file
    try:
        # Load + Process + data cleanup
        df = load and clean dataset(filepath)
        df = engineer features data_prep(df)
        # Exploratory Data Analysis
        run Exploratory_Data_Analysis(df)
        # Modeling
        (X train, X test, y_train, y_test), features =
prepare model data(df)
        model = train_predictive_model(X_train, y_train, X_test,
y test)
        runConfusionMatrix(X_test, model, y_test)
        # Routing Simulation
        psp models = train routing models(df, features)
        # Pick a sample and simulate routing
        if not X test.empty:
            sample tx = X test.iloc[[0]]  # First row of test set
            simulate_routing_with_cost(psp_models, sample_tx,
psp fees)
        else:
            print(" No test data available to simulation routing.")
    except Exception as e:
        print(f"\n Unexpected error in main: {str(e)}")
        sys.exit(1)
```

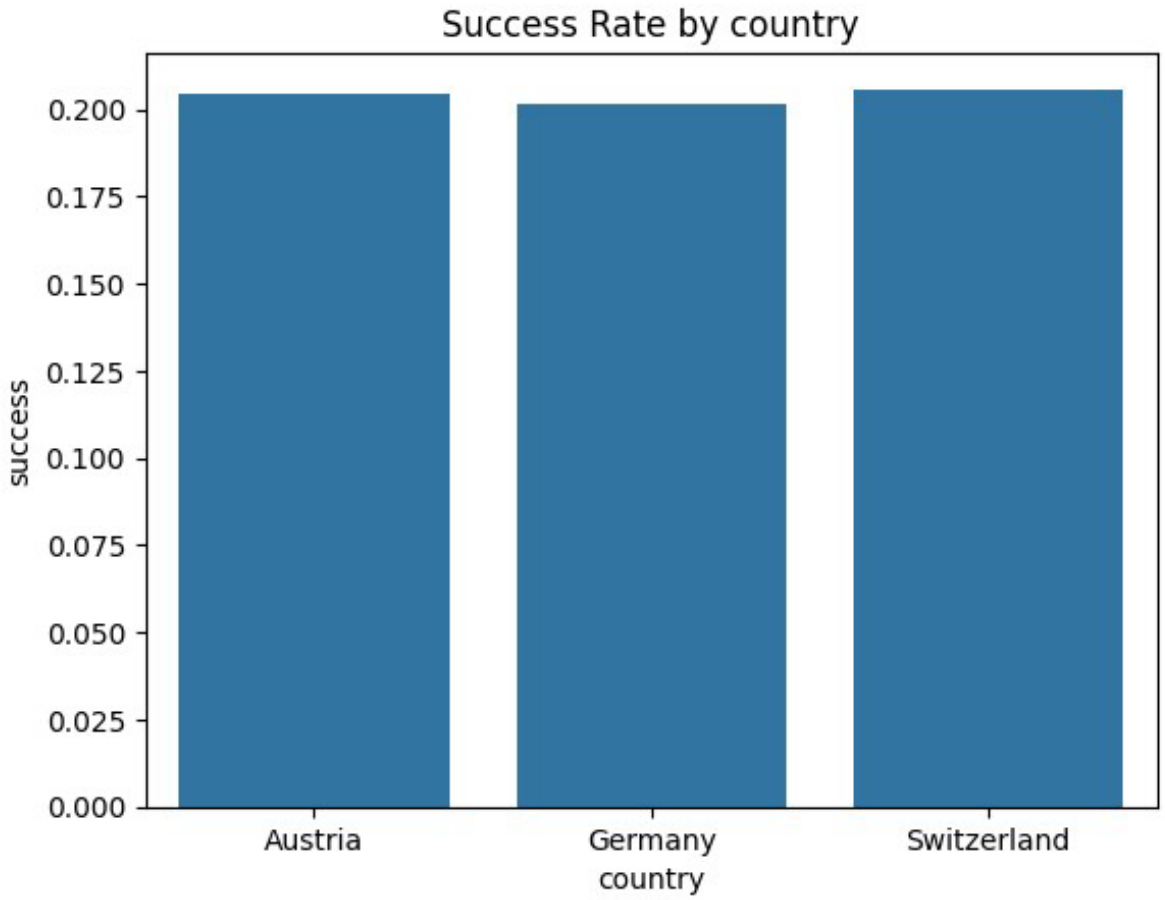
Output:

```
File read and cleaned successfully.

Feature engineering completed.

Success rate per PSP:
PSP
Goldcard      0.406172
Moneycard     0.218754
Simplecard    0.158123
UK_Card       0.194338
Name: success, dtype: float64

Success rate by 3D Secure flag:
3D secured
0    0.189562
1    0.245525
Name: success, dtype: float64
```

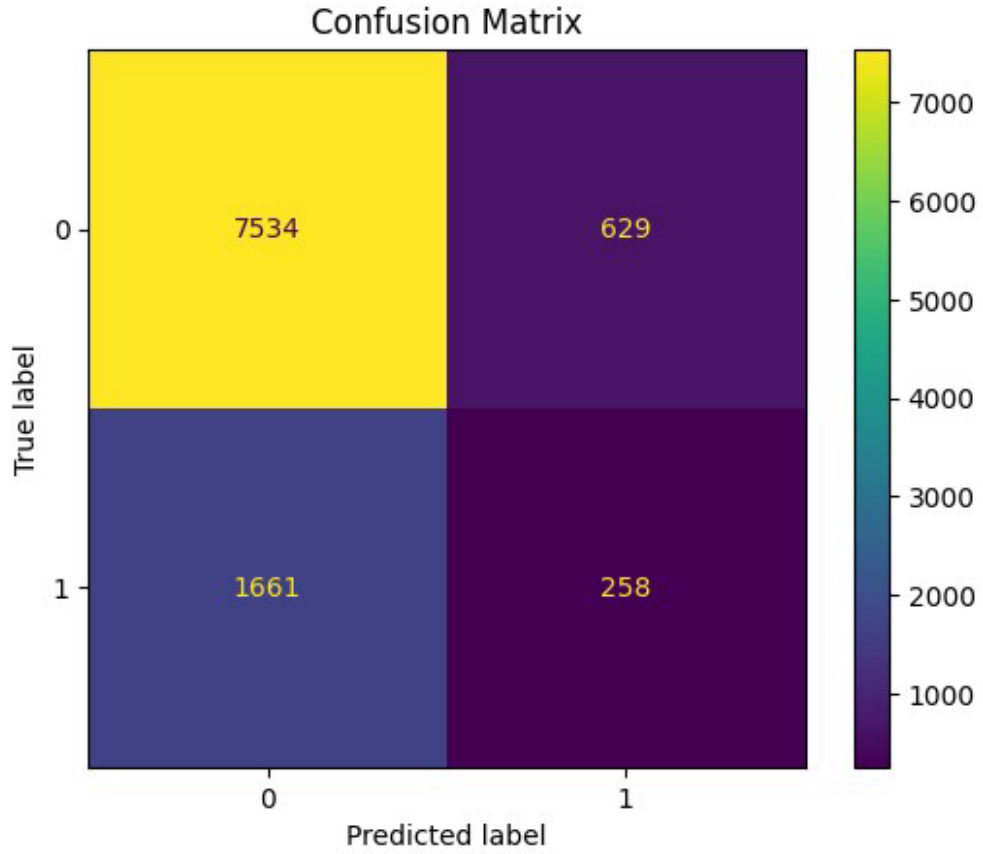



Success Rate by PSP
success
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
UK_Card
Simplecard
Moneycard
Goldcard
PSP


Success Rate by hour
success
0.00
0.05
0.10
0.15
0.20
0.25
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
hour




```
Model prepared!

Model Performance:
              precision    recall  f1-score   support

           0       0.82      0.92      0.87      8163
           1       0.29      0.13      0.18      1919

    accuracy                           0.77     10082
   macro avg       0.56      0.53      0.53     10082
weighted avg       0.72      0.77      0.74     10082
```



	precision	recall	f1-score	support
0	0.82	0.92	0.87	8163
1	0.29	0.13	0.18	1919
accuracy			0.77	10082
macro avg	0.56	0.53	0.53	10082
weighted avg	0.72	0.77	0.74	10082

```
UK Card: Expected cost: €1.10
Simplecard: Expected cost: €0.50
Moneycard: Expected cost: €2.00
Goldcard: Expected cost: €5.00

PSP Scores (Success Probability and Expected Cost):
UK Card: P(success)=0.05, Expected Cost=1.1000 €
Simplecard: P(success)=0.00, Expected Cost=0.5000 €
Moneycard: P(success)=0.00, Expected Cost=2.0000 €
Goldcard: P(success)=0.00, Expected Cost=5.0000 €

Optimal PSP (lowest expected cost): Simplecard with cost 0.5000 €
```